

Student name: Lê Ngọc Anh Vũ

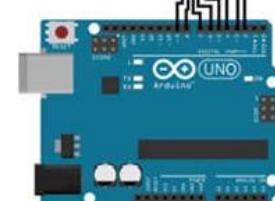
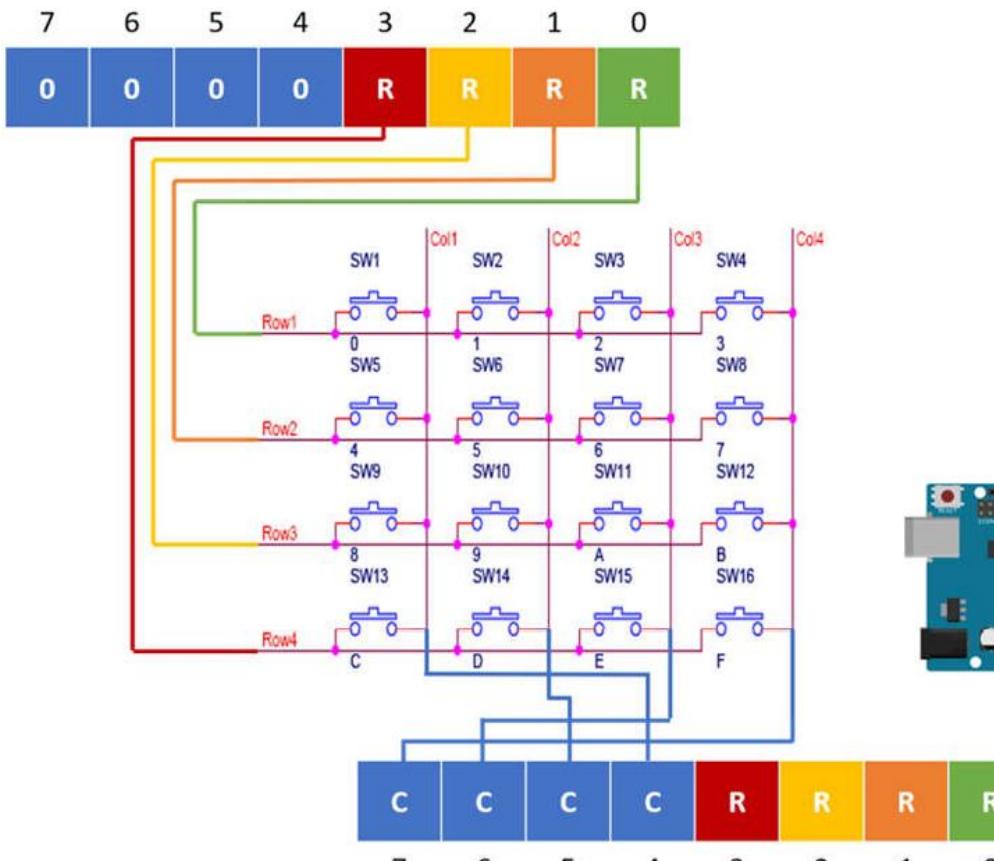
Student ID: 20236014

Lab 11

Assignment 1: Create a new project, type in, and build the program of Home Assignment 1. Run the program step by step to understand each line of the source code. Upgrade the source code so that it could detect all 16 key buttons, from 0 to F.

IN_ADDRESS_HEXA_KEYBOARD

Address 0xFFFF0012

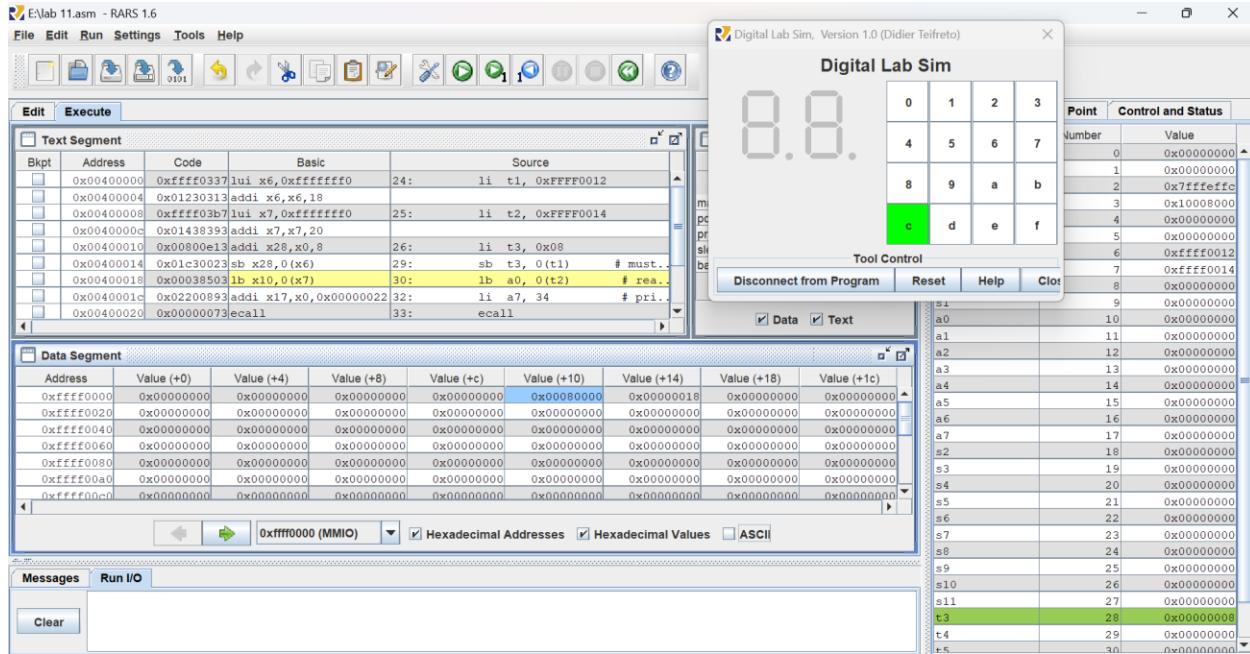


OUT_ADDRESS_HEXA_KEYBOARD

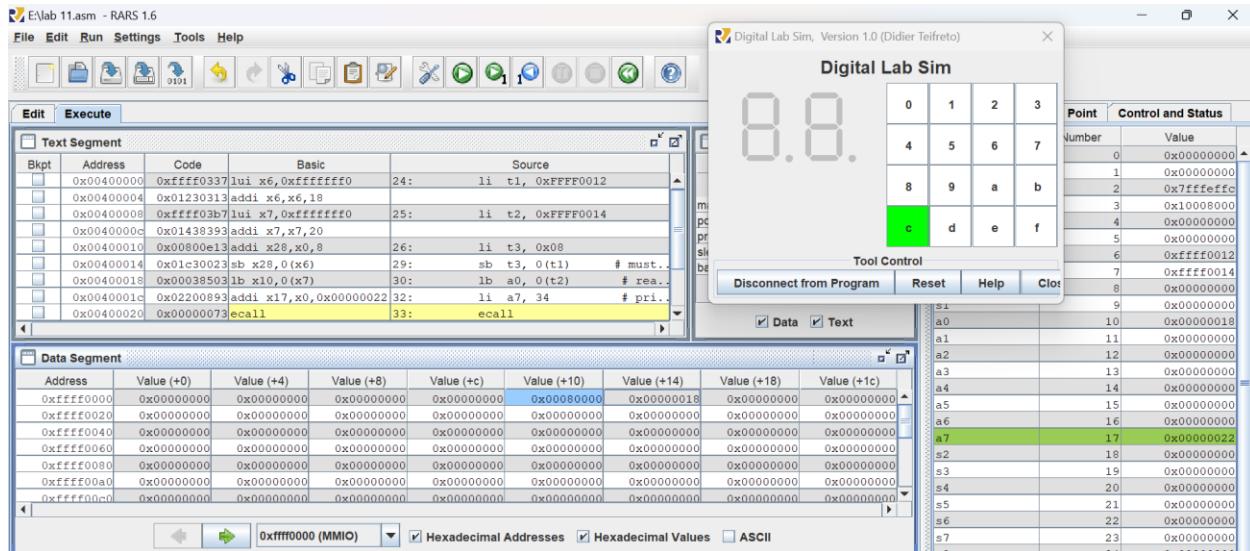
Address 0xFFFF0014

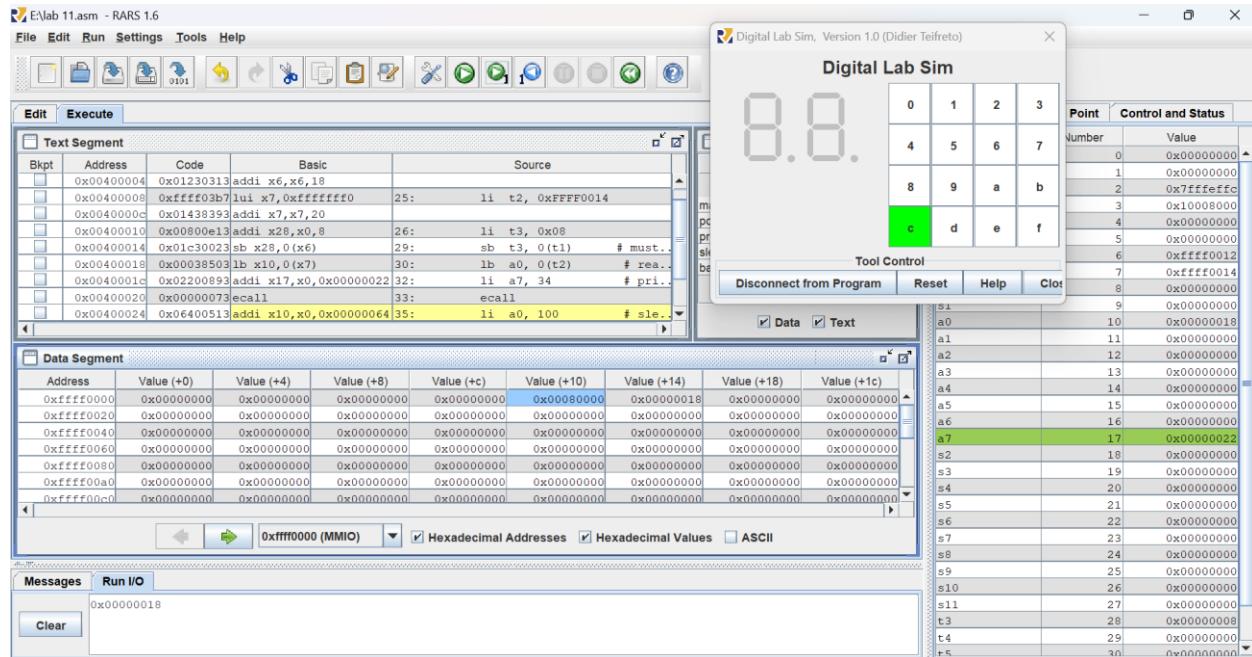
Run the program step by step:

Assigning value of row4 for the address 0xFFFF0012 which is 00001000 in binary or 0x8 in hexadecimal.

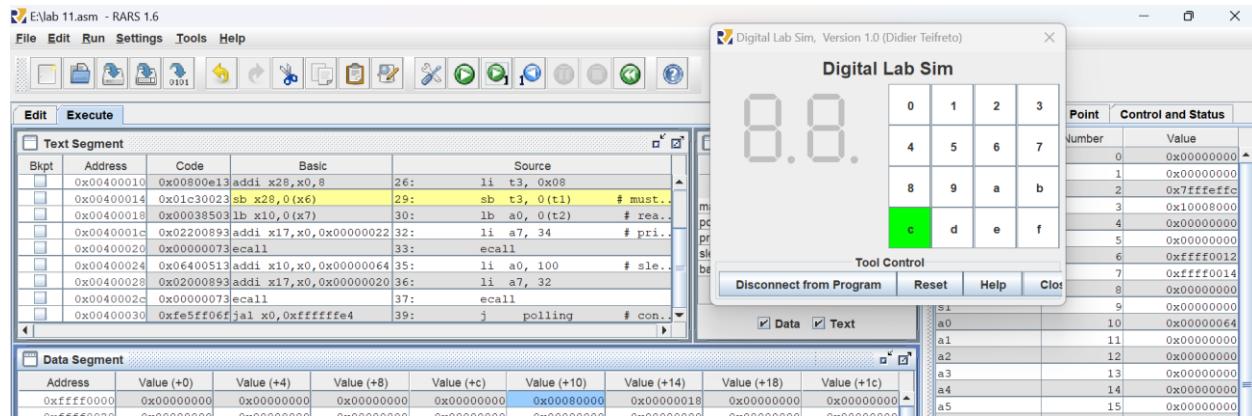


At the present, the address 0xFFFF0014 stores value of button key which is pressed. We load this value to the register t2 and print it to console

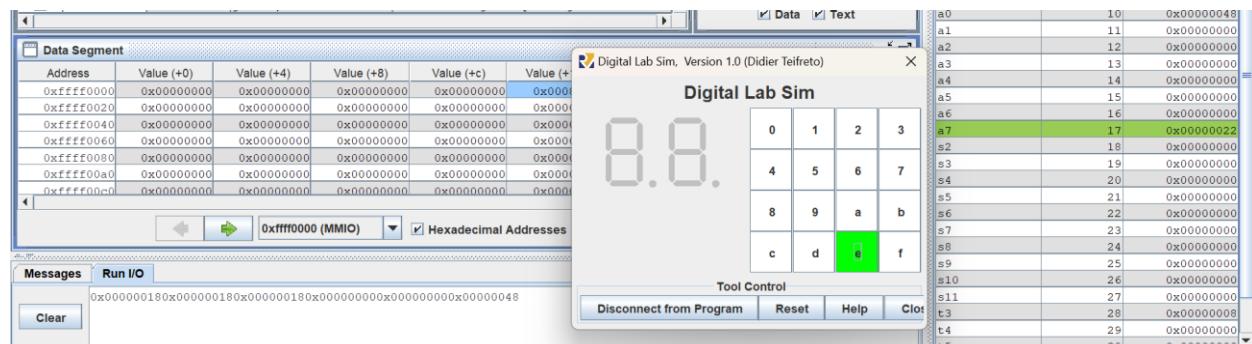




After printing the key number to console , the program sleeps in 100 ms. Then, it does the loop with the instruction “j polling” and executes above process again.



The value printing to console will have not changed until you press other button in set {d, e, f} or do not press any button (the value will be 0).



The updated source code is as below:

Editor window showing the assembly code for lab 11.asm. The code defines memory locations for a 4x4 keyboard matrix and handles key presses.

```
1 # -----  
2 #      col 0x1      col 0x2      col 0x4      col 0x8  
3 #  row 0x1      0      1      2      3  
4 #          0x11      0x21      0x41      0x81  
5 #  row 0x2      4      5      6      7  
6 #          0x12      0x22      0x42      0x82  
7 #  row 0x4      8      9      a      b  
8 #          0x14      0x24      0x44      0x84  
9 #  row 0x8      c      d      e      f  
10 #         0x18      0x28      0x48      0x88  
11 # -----  
12 # Command row number of hexadecimal keyboard (bit 0 to 3)  
13 # Eg. assign 0x1, to get key button 0,1,2,3  
14 #      assign 0x2, to get key button 4,5,6,7  
15 # NOTE must reassign value for this address before reading,  
16 # even though you only want to scan 1 row  
17 .eqv IN_ADDRESS_HEXA_KEYBOARD      0xFFFF0012  
18 # Receive row and column of the key pressed, 0 if not key pressed  
19 # Eg. equal 0x11, means that key button 0 pressed.  
20 # Eg. equal 0x28, means that key button D pressed.  
21 .eqv OUT_ADDRESS_HEXA_KEYBOARD     0xFFFF0014  
22 .text
```

Line: 53 Column: 7 Show Line Numbers

Editor window showing the assembly code for lab 11.asm. The code includes a main loop for polling the keyboard.

```
19 # Eg. equal 0x11, means that key button 0 pressed.  
20 # Eg. equal 0x28, means that key button D pressed.  
21 .eqv OUT_ADDRESS_HEXA_KEYBOARD     0xFFFF0014  
22 .text  
23 main:  
24     li t1, IN_ADDRESS_HEXA_KEYBOARD  
25     li t2, OUT_ADDRESS_HEXA_KEYBOARD  
26 # check row 4 with key C, D, E, F  
27 polling:  
28     li t3, 0x01  
29     sb t3, 0(t1)      # must reassign expected row  
30     lb a0, 0(t2)      # read scan code of key button  
31     bnez a0, print  
32  
33     li t3, 0x02  
34     sb t3, 0(t1)  
35     lb a0, 0(t2)  
36     bnez a0, print  
37  
38     li t3, 0x04  
39     sb t3, 0(t1)  
40     lb a0, 0(t2)
```

Line: 25 Column: 36 Show Line Numbers

Edit Execute

lab 11.asm

```

39      sb t3, 0(t1)
40      lb a0, 0(t2)
41      bnez a0, print
42
43      li t3, 0x08
44      sb t3, 0(t1)
45      lb a0, 0(t2)
46      bnez a0, print
47  print:
48      li a7, 34          # print integer (hexa)
49      ecall
50
51      li a7, 11
52      li a0, 32
53      ecall
54  sleep:
55      li a0, 100         # sleep 100ms
56      li a7, 32
57      ecall
58  back_to_polling:
59      j polling         # continue polling

```

Line: 53 Column: 7 Show Line Numbers

The new program will consider each row to detect which button key is pressed. If we have the answer, print the number key. If there are not any buttons are pressed, print 0x00000000

Example:

Digital Lab Sim, Version 1.0 (Didier Telfret)

Digital Lab Sim

0	1	2	3
4	5	6	7
8	9	a	b
c	d	e	f

Tool Control: Disconnect from Program, Reset, Help, Close

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)
0xfffff0000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0xfffff0020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0xfffff0040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0xfffff0060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0xfffff0080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0xfffff00a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0xfffff00c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

Messages Run I/O

0x00000000

Clear

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)
0xfffff0000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0xfffff0020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0xfffff0040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0xfffff0060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0xfffff0080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0xfffff00a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0xfffff00c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

Messages Run I/O

0x00000000 0x00000000 0xffffffff81

Clear

Digital Lab Sim, Version 1.0 (Didier Telfret)

Digital Lab Sim

0	1	2	3
4	5	6	7
8	9	a	b
c	d	e	f

Tool Control: Disconnect from Program, Reset, Help, Close

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)
0xfffff0000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0xfffff0020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0xfffff0040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0xfffff0060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0xfffff0080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0xfffff00a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0xfffff00c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

Messages Run I/O

0x00000000 0x00000000 0xffffffff81

Clear

Digital Lab Sim, Version 1.0 (Didier Telfreto)

Digital Lab Sim

8.8.

0	1	2	3
4	5	6	7
8	9	a	b
c	d	e	f

Tool Control

Disconnect from Program Reset Help Close

7 0xfffff0014
8 0x00000000
9 0x00000000
10 0x00000020
11 0x00000000
12 0x00000000
13 0x00000000
14 0x00000000
15 0x00000000
16 0x00000000
17 0x0000000b
18 0x00000000
19 0x00000000
20 0x00000000
21 0x00000000
22 0x00000000
23 0x00000000
24 0x00000000
s9 25 0x00000000
s10 26 0x00000000
s11 27 0x00000000
t3 28 0x00000004
t4 29 0x00000000
r5 30 0x00000000

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)
0xfffff0000	0x00000000	0x00000000	0x00000000	0x00000000	0x00004000	0x00000000
0xfffff0020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0xfffff0040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0xfffff0060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0xfffff0080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0xfffff00a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0xfffff00c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

0xfffff0000 (MMIO) Hexadecimal Addresses Hexadecimal Values

Messages Run I/O

0x00000000 0x00000000 0xfffffff81 0x00000024

Clear

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)
0xfffff0000	0x00000000	0x00000000	0x00000000	0x00000000	0x00020000	0x00000012
0xfffff0020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0xfffff0040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0xfffff0060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0xfffff0080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0xfffff00a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0xfffff00c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

0xfffff0000 (MMIO) Hexadecimal Addresses Hexadecimal Values

Messages Run I/O

0x00000000 0x00000000 0xfffffff81 0x00000024 0x00000012

Clear

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)
0xfffff0000	0x00000000	0x00000000	0x00000000	0x00000000	0x00080000	0x00000048
0xfffff0020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0xfffff0040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0xfffff0060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0xfffff0080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0xfffff00a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0xfffff00c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

0xfffff0000 (MMIO) Hexadecimal Addresses Hexadecimal Values

Messages Run I/O

0x00000000 0x00000000 0xfffffff81 0x00000024 0x00000012 0x00000048

Clear

Assignment 2: Create a new project, type in, and build the program of Home Assignment 2. Run the program step by step to understand each line of the source code.

Load address of handler to register t0

Edit Execute

Text Segment

Blkt	Address	Code	Basic	Source
0x00400000	0x000000297	auipc x5,0	10:	la t0, handler
0x00400004	0x04028293	addi x5,x5,0x00000040		
0x00400008	0x0052a073	csrrs x0,5,x5	11:	csrrs zero, utvec, t0
0x00400006	0x10000313	addi x6,x0,0x00000100	14:	li t1, 0x100
0x00400010	0x0432073	csrrs x0,4,x6	15:	csrrs zero, uie, t1 #..
0x00400014	0x0000e073	csrrsi x0,0,1	17:	csrrsi zero, ustatus, 0x1 #..
			20:	li t1, 0xFFFF0012
			21:	li t3, 0x80 # bit 7 = 1..

Digital Lab Sim

Labels

Label	Address
lab 11.asm	0x00400000
main	0x00400000
loop	0x00400028
end_main	0x00400040
handler	0x00400040
message	0x10010000

Registers

Name	Number	Value
zero	0	0x00000000
r1	1	0x00000000
sp	2	0x7fffffc
gp	3	0x10008000
tp	4	0x00000000
t0	5	0x00400040
t1	6	0x00000000
t2	7	0x00000000
s0	8	0x00000000
s1	9	0x00000000
a0	10	0x00000000
a1	11	0x00000000

Then, assign this address to utvec

Name	Number	Value
ustatus	0	0x00000000
fflags	1	0x00000000
frm	2	0x00000000
fcsr	3	0x00000000
kie	4	0x00000000
utvec	5	0x00400040
uscratch	64	0x00000000
uepc	65	0x00000000
ucause	66	0x00000000
utval	67	0x00000000
uip	68	0x00000000
cycle	3072	0x00000003
time	3073	0x76008d0b

Set the UEIE bit in UIE register

Name	Number	Value
ustatus	0	0x00000000
fflags	1	0x00000000
frm	2	0x00000000
fcsr	3	0x00000000
kie	4	0x00000100
utvec	5	0x00400040
uscratch	64	0x00000000
uepc	65	0x00000000
ucause	66	0x00000000
utval	67	0x00000000
uip	68	0x00000000
cycle	3072	0x00000005
time	3073	0x760232da
instret	3074	0x00000005
cycleh	3200	0x00000000
timet	3201	0x00000193
instreth	3202	0x00000000

Set the UIE bit in USTATUS register in order to enable interrupts

Name	Number	Value
ustatus	0	0x00000001
fflags	1	0x00000000
frm	2	0x00000000
fcsr	3	0x00000000
kie	4	0x00000100
utvec	5	0x00400040
uscratch	64	0x00000000
uepc	65	0x00000000
ucause	66	0x00000000
utval	67	0x00000000
uip	68	0x00000000
cycle	3072	0x00000006
time	3073	0x7604db31
instret	3074	0x00000006
cycleh	3200	0x00000000
timet	3201	0x00000193
instreth	3202	0x00000000

Enable the interrupt of keypad of Digital Lab Sim, set bit 7 to value of 1

Name	Number	Value
t2	7	0x00000000
s0	8	0x00000000
s1	9	0x00000000
a0	10	0x00000000
a1	11	0x00000000
a2	12	0x00000000
a3	13	0x00000000
a4	14	0x00000000
a5	15	0x00000000
a6	16	0x00000000
a7	17	0x00000000
s2	18	0x00000000
s3	19	0x00000000
s4	20	0x00000000
s5	21	0x00000000
s6	22	0x00000000
s7	23	0x00000000
s8	24	0x00000000
s9	25	0x00000000
s10	26	0x00000000
s11	27	0x00000000
t3	28	0x00000000

At this time, we have already finished all of preparation and just wait for interrupts occurring.

The program will have executed this loop until there exists interrupts. In this case, interrupt is pressing an button. Then, the main program will stops temporarily to handle interrupts. The context in main program is stored in stack and restored after the interrupts have been handled. After that, handler prints message "Someone's presed a button."

Digital Lab Sim, Version 1.0 (Didier Telfret)

Digital Lab Sim

8	.	8	
0	1	2	3
4	5	6	7
8	9	a	b
c	d	e	f

Tool Control

Disconnect from Program Reset Help Close Hexadecimal Addresses Hexadecimal Values ASCII

Registers

Name	Number	Value
zero	0	0x00000000
ra	1	0x00000000
sp	2	0x7ffffefffc
gp	3	0x10000000
tp	4	0x00000000
t0	5	0x00400040
t1	6	0xfffff0012
t2	7	0x00000000
s0	8	0x00000000
s1	9	0x00000000
a0	10	0x00000000
a1	11	0x00000000
a2	12	0x00000000
a3	13	0x00000000
a4	14	0x00000000
a5	15	0x00000000
a6	16	0x00000000
a7	17	0x00000000
s2	18	0x00000000
s3	19	0x00000000
s4	20	0x00000000
s5	21	0x00000000
s6	22	0x00000000
s7	23	0x00000000
s8	24	0x00000000

Digital Lab Sim, Version 1.0 (Didier Telfret)

Digital Lab Sim

8	.	8	
0	1	2	3
4	5	6	7
8	9	a	b
c	d	e	f

Tool Control

Disconnect from Program Reset Help Close Hexadecimal Addresses Hexadecimal Values ASCII

Registers

Name	Number	Value
zero	0	0x00000000
ra	1	0x00000000
sp	2	0x7ffffeffe4
gp	3	0x10000000
tp	4	0x00000000
t0	5	0x00400040
t1	6	0xfffff0012
t2	7	0x00000000
s0	8	0x00000000
s1	9	0x00000000
a0	10	0x00000000
a1	11	0x00000000
a2	12	0x00000000
a3	13	0x00000000
a4	14	0x00000000
a5	15	0x00000000
a6	16	0x00000000
a7	17	0x00000004
s2	18	0x00000000
s3	19	0x00000000
s4	20	0x00000000
s5	21	0x00000000
s6	22	0x00000000
s7	23	0x00000000
s8	24	0x00000000

Digital Lab Sim, Version 1.0 (Didier Telfret)

Digital Lab Sim

8	.	8	
0	1	2	3
4	5	6	7
8	9	a	b
c	d	e	f

Tool Control

Disconnect from Program Reset Help Close Hexadecimal Addresses Hexadecimal Values ASCII

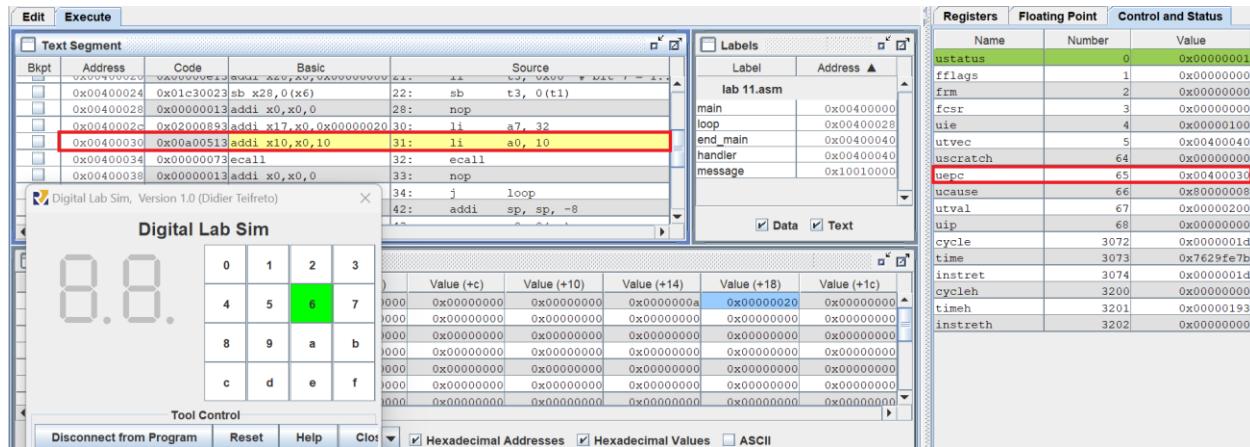
Messages

Someone's presed a button.

Registers

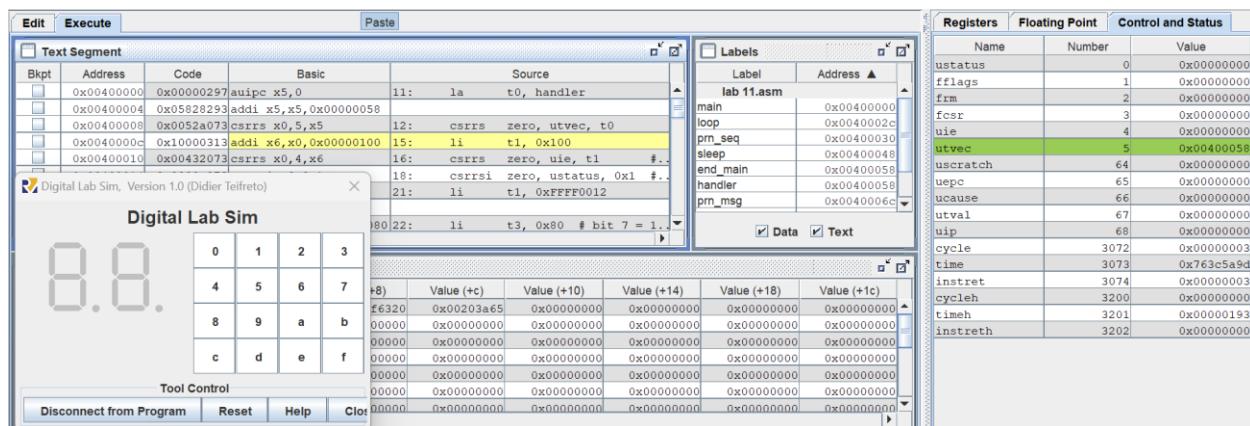
Name	Number	Value
zero	0	0x00000000
ra	1	0x00000000
sp	2	0x7ffffeffe4
gp	3	0x10000000
tp	4	0x00000000
t0	5	0x00400040
t1	6	0xfffff0012
t2	7	0x00000000
s0	8	0x00000000
s1	9	0x00000000
a0	10	0x10010000
a1	11	0x00000000
a2	12	0x00000000
a3	13	0x00000000
a4	14	0x00000000
a5	15	0x00000000
a6	16	0x00000000
a7	17	0x00000004
s2	18	0x00000000
s3	19	0x00000000
s4	20	0x00000000
s5	21	0x00000000
s6	22	0x00000000
s7	23	0x00000000
s8	24	0x00000000
s9	25	0x00000000
s10	26	0x00000000
t3	27	0x00000000
t4	28	0x00000080
r	29	0x00000000
ra	30	0x00000000

Finally, return to the main program, come back to the instruction which is about to be executed when interrupts occur that its address stored in uepc register. Then, the main program continues to do the loop until interrupts occur again.

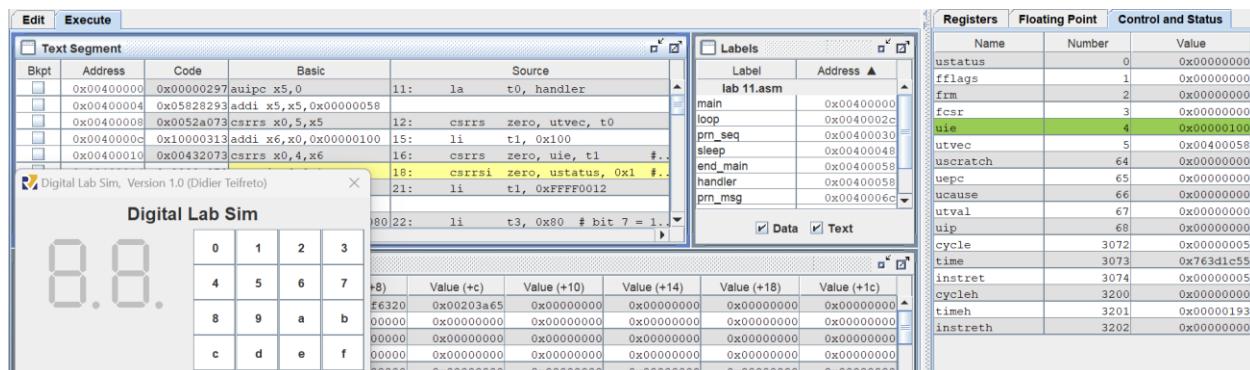


Assignment 3: Create a new project, type in, and build the program of Home Assignment 3. Run the program step by step to understand each line of the source code. Upgrade the source code so that it could detect all 16 key buttons, from 0 to F.

Then, assign this address to utvec



Set the UEIE bit in UIE register (bit 8)



Set the UIE bit (bit 0) in USTATUS register in order to enable interrupts

The screenshot shows the Digital Lab Sim interface. On the left, the assembly code for lab 11.asm is displayed, showing instructions like `csrrs zero, utvec, t0` and `csrrsi zero, ustatus, 0x1`. In the center, a digital display shows the value 8.8. On the right, the Registers window shows the USTATUS register at address 0x00000000 with its value set to 1. The floating point and control/status registers are also visible.

Enable the interrupt of keypad of Digital Lab Sim, set bit 7 to value of 1

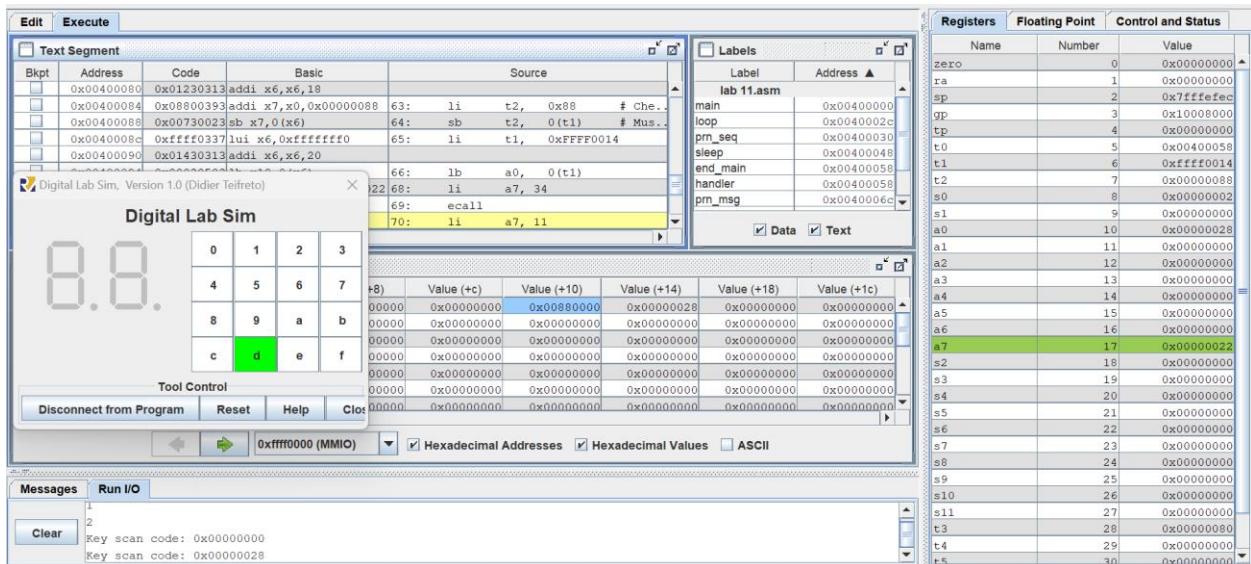
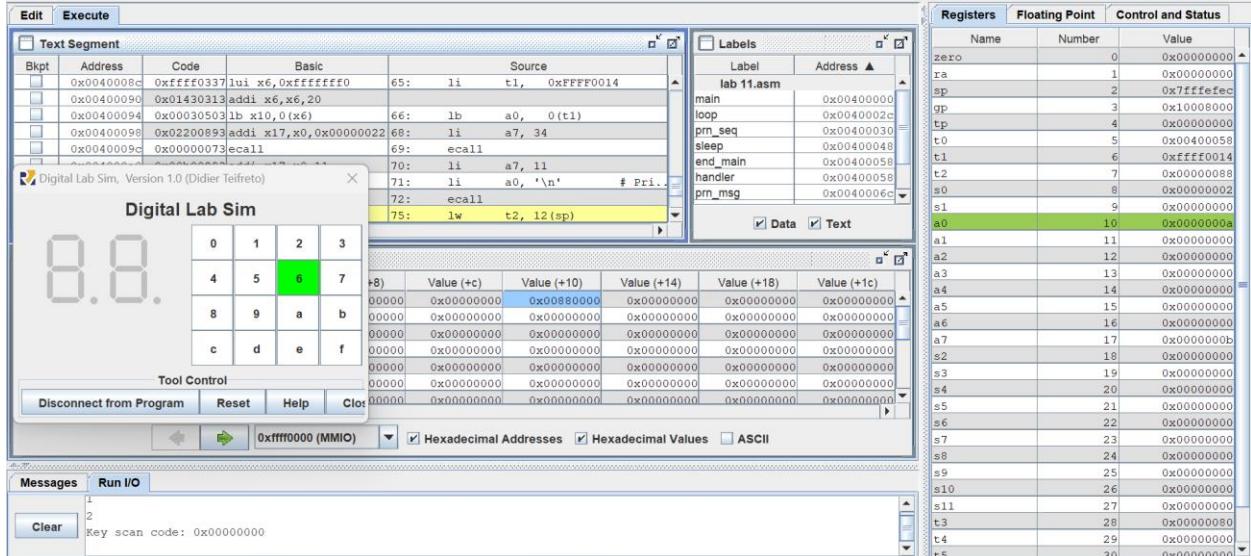
This screenshot shows the assembly code with a modification: the instruction `xor s0, s0, s0` has been added at address 0x00040018. The digital display still shows 8.8. The Registers window now shows the USTATUS register at address 0x00000000 with its value set to 0. The floating point and control/status registers are also visible.

At this time, we have already finished all of preparation and just wait for interrupts occurring.

The main program will have executed the loop and printed sequence of number starting at 1 until interrupts occur.

This screenshot shows the assembly code with the modification. The digital display shows 8.8. The Registers window shows the USTATUS register at address 0x00000000 with its value set to 0. The floating point and control/status registers are also visible. The bottom part of the interface shows the messages window with entries '1' and '2'.

The main program stops temporarily in order to handle the interrupts. Before handling, we store the current context in stack and restore it after interrupts have already handled. In this case, the interrupts are pressing button. If user presses any buttons in row 4, handler prints the number key of that button. If user presses other buttons which are not in row4, handler prints value of 0x00000000.



After handling the interrupts, return to the main program, continue to do the loop and wait for the next interrupts.

The updated source code is as below:

Edit Execute

lab 11.asm

```
1 .eqv IN_ADDRESS_HEXA_KEYBOARD      0xFFFFF0012
2 .eqv OUT_ADDRESS_HEXA_KEYBOARD    0xFFFFF0014
3 .data
4     message: .asciz "Key scan code: "
5 # -----
6 # MAIN Procedure
7 # -----
8 .text
9 main:
10    # Load the interrupt service routine address to the UTVEC register
11    la      t0, handler
12    csrrs  zero, utvec, t0
13
14    # Set the UEIE (User External Interrupt Enable) bit in UIE register
15    li      t1, 0x100
16    csrrs  zero, uie, t1      # uie - ueie bit (bit 8)
17    # Set the UIE (User Interrupt Enable) bit in USTATUS register
18    csrrsi zero, ustatus, 0x1 # ustatus - enable uie (bit 0)
19
20    # Enable the interrupt of keypad of Digital Lab Sim
21    li      t1, IN_ADDRESS_HEXA_KEYBOARD
22    li      t3, 0x80 # bit 7 = 1 to enable interrupt
```

Edit Execute

lab 11.asm

```
21    li      t1, IN_ADDRESS_HEXA_KEYBOARD
22    li      t3, 0x80 # bit 7 = 1 to enable interrupt
23    sb      t3, 0(t1)
24
25    # -----
26    # Loop to print a sequence numbers
27    # -----
28    xor     s0, s0, s0      # count = s0 = 0
29 loop:
30    addi   s0, s0, 1       # count = count + 1
31 prn_seq:
32    addi   a7, zero, 1
33    add    a0, s0, zero    # Print auto sequence number
34    ecall
35    addi   a7, zero, 11
36    li     a0, '\n'        # Print EOL
37    ecall
38 sleep:
39    addi   a7, zero, 32
40    li     a0, 300         # Sleep 300 ms
41    ecall
42    j     loop
```

Line: 62 Column: 41 Show Line Numbers

Edit Execute

lab 11.asm

```
42     j      loop
43 end_main:
44
45 # -----
46 # Interrupt service routine
47 #
48 handler:
49     # Saves the context
50     addi    sp, sp, -16
51     sw     a0, 0(sp)
52     sw     a7, 4(sp)
53     sw     t1, 8(sp)
54     sw     t2, 12(sp)
55
56     # Handles the interrupt
57 prn_msg:
58     addi    a7, zero, 4
59     la     a0, message
60     ecall
61 get_key_code:
62     li     t1, IN_ADDRESS_HEXA_KEYBOARD
63     li     t2, 0x81
||
```

Line: 62 Column: 41 Show Line Numbers

Edit Execute

lab 11.asm

```
63     li     t2, 0x81
64     sb     t2, 0(t1)      # must reassign expected row
65     li     t1, OUT_ADDRESS_HEXA_KEYBOARD
66     lb     a0, 0(t1)      # read scan code of key button
67     bneq a0, prn_key_code
68
69     li     t1, IN_ADDRESS_HEXA_KEYBOARD
70     li     t2, 0x82
71     sb     t2, 0(t1)
72     li     t1, OUT_ADDRESS_HEXA_KEYBOARD
73     lb     a0, 0(t1)
74     bneq a0, prn_key_code
75
76     li     t1, IN_ADDRESS_HEXA_KEYBOARD
77     li     t2, 0x84
78     sb     t2, 0(t1)
79     li     t1, OUT_ADDRESS_HEXA_KEYBOARD
80     lb     a0, 0(t1)
81     bneq a0, prn_key_code
82
83     li     t1, IN_ADDRESS_HEXA_KEYBOARD
84     li     t2, 0x88
||
```

Line: 62 Column: 41 Show Line Numbers

Edit Execute

Lab 11.asm

```

44      li t2, 0x88
45      sb t2, 0(t1)
46      li t1, OUT_ADDRESS_HEXA_KEYBOARD
47      lb a0, 0(t1)
48      bnez a0, prn_key_code
49
50      .prn_key_code:
51      li a7, 34
52      ecall
53      li a7, 11
54      li a0, '\n'          # Print EOL
55      ecall
56
57      # Restores the context
58      lw t2, 12(sp)
59      lw t1, 8(sp)
60      lw a7, 4(sp)
61      lw a0, 0(sp)
62      addi sp, sp, 16
63
64      # Back to the main procedure
65      uret

```

Line: 62 Column: 41 Show Line Numbers

The result is:

